b)

Original document

SIMULATION DEVICE FOR ASSISTING SOFTWARE DEVELOPMENT

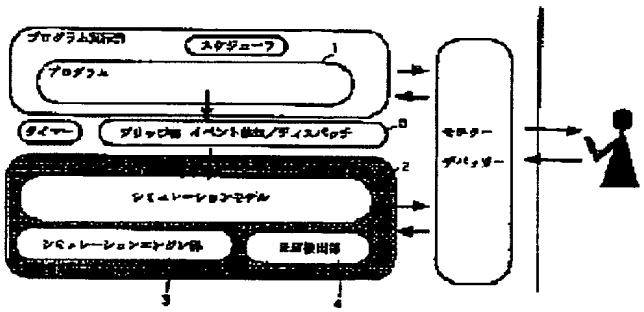
Patent number: JP9305389
Publication date: 1997-11-28
Inventor: ODA TOSHIHIKO
Applicant: RICOH KK
Classification:
- international: ***G06F9/06; G06F9/44; G06F11/28;*** **G06F9/06; G06F9/44; G06F11/28;** (IPC1-7): G06F9/06; G06F9/44; G06F11/28
- european:
Application number: JP19960145018 19960515
Priority number(s): JP19960145018 19960515

View INPADOC patent family

Report a data error here

Abstract of **JP9305389**

PROBLEM TO BE SOLVED: To create development environment wherein a simulation model and a development program are linked. SOLUTION: A program 1 to be developed and the simulation model 2 are placed in operated at the same time and the behavior of a system is confirmed to obtain more accurate values of various parameters determining the operation of the system based upon the program 1. The simulation model 2 simulate the behavior of a machine, controlled with commands from the program 1, on a computer, and consists of constituent elements called devices. A contradiction detection part 4 is actuated immediately once a simulation engine 3 completes the update of the model 2 to decide whether or not the states of all the devices meet the specifications of the system and informs a user of a device which does not meet the specification. The execution of the program 1 is the execution of successive procedures, the execution of the simulation model 2 is nonsuccessive state transition type execution, and a bridge part 5 compensates their differences in execution form to enable them to communicate information with each other.



Data supplied from the *esp@cenet* database - Worldwide

b)

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－305389

(43)公開日　平成９年(1997)11月28日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	9/06	５３０		Ｃ０６Ｆ	9/06	５３０Ｔ
	9/44	５３０			9/44	５３０Ｚ
	11/28	３４０			11/28	３４０Ｃ

審査請求　未請求　請求項の数４　ＦＤ　（全８頁）

(21)出願番号　特願平8－145018

(22)出願日　平成８年(1996)５月15日

(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号

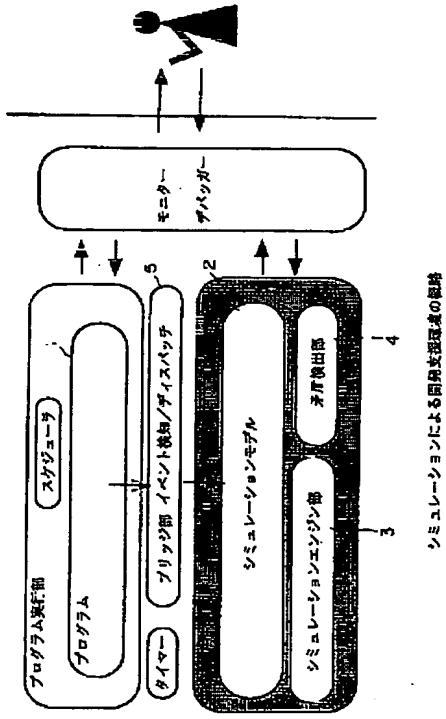
(72)発明者　小田　利彦
東京都大田区中馬込１丁目３番６号　株式会社リコー内

(54)【発明の名称】　ソフトウエア開発支援のシミュレーション装置

(57)【要約】

【課題】　シミュレーションモデルと開発プログラムがリンクされた開発環境を作る。

【解決手段】　開発対象プログラム１とシミュレーションモデル２を同時に動作をさせてシステムの挙動を確認し、プログラム上のシステムの動作を決定する各種のパラメータのより正確な値を得る。モデル２は、プログラム１からの指令によって制御される機械の挙動を計算機上で模擬するもので、デバイスとよぶ構成要素から構成されている。矛盾検出部４は、シミュレーションエンジン３がモデルの更新を終了すると直ちに起動されて、全てのデバイスの状態がシステムの仕様を満たしているかどうかを判定し、満たしていないとそのデバイスをユーザに知らせる。プログラムの実行は連続的な手続きの実行であり、シミュレーションモデルの実行は非連続で状態遷移型の実行であり、ブリッジ部５は、これらの実行形態の違いを埋め、互いに情報の通信が行えるようにする。



【特許請求の範囲】
【請求項1】　プログラムからの制御に基づき制御対象の物理的対象の挙動をシミュレーションするソフトウエア開発支援のシミュレーション装置において、シミュレーションモデルは、個々の物理的対象を表現するデバイスと呼ぶ要素から構成され、デバイスには、デバイスが持つ特性や性質の情報（属性と呼ぶ）、デバイスの取りうる状態とその状態遷移に関する情報、さらに、各状態毎に、その状態が存在を判定する論理的条件（成立条件と呼ぶ）、その状態内で成立する属性の関数的記述（状態関数と呼ぶ）が定義されていることを特徴とするソフトウエア開発支援のシミュレーション装置。
【請求項2】　デバイス間の属性の依存関係をパイプと呼ぶ機能で実現し、パイプは、あるデバイスの属性と別のデバイスの属性との対応関係とそれらがどのような依存関係にあるかを示す関数的記述から成ることを特徴とするソフトウエア開発支援のシミュレーション装置。
【請求項3】　ブリッジ部に、シミュレーションモデルの状態変化とプログラムのイベントとの対応関係を定義するテーブルが存在し、該ブリッジ部は、プログラム実行部内で実行中のプログラムが送るデバイスへの指令に基づき、シミュレーション部内のデバイスの状態を遷移させる機能とシミュレーション部内のデバイスの状態変化を検出し、プログラム側にイベントのディスパッチを行う機能を実行することを特徴とするソフトウエア開発支援のシミュレーション装置。
【請求項4】　開発支援環境は、シミュレーションモデルがシステムの動作仕様と矛盾する状態を検知してそれを利用者に通知する矛盾検知部を持ち、矛盾検知部は、シミュレーションの実行中に各デバイスの状態を調べて、与えられたシステム動作仕様の記述に基づき、デバイスの異常状態を検出し、それをモニタ上に表示することを特徴とするソフトウエア開発支援のシミュレーション装置。
【発明の詳細な説明】
【0001】
【発明の属する技術分野】本発明は、機械制御型ソフトウエアの開発支援のシミュレーション装置に関する。
【0002】
【従来の技術】本発明は、機械の動作を制御するソフトウエアの開発効率を向上させる開発環境を提供するために、機械の動作を模擬するシミュレータをプログラムの実行とリンクさせて、動作の確認や不具合の発見を容易にすることを目的とするものであるが、その背景問題として、
1．機械動作のタイミングや動作時間等はタイミングチャートでソフトウエアに仕様が与えられるが、そのとおりにプログラムを作成しても、実際にハードウエアを動かしてみるとシステムの仕様通りには動作しないことが多い。
2．ハードウエアが出来上がっていないとソフトウエアのテストや検証が行えない。等があげられる。
【0003】特開平7－84832号公報（リアルタイムシミュレーション装置）の発明は、オブジェクト間のメッセージの送受信によってシミュレーションを実行するもので、メッセージパスを格納して、解釈実行するものである。
【0004】特開平4－226426号公報（情報処理装置のシミュレータ及びシミュレーション装置）の発明は、情報処理装置を模擬するシミュレータでデバイス制御を模擬するデバイス装置模擬機能を、モデルを表示する機能とデバイスの操作に応じて表示モデルの動作を模擬するモデル模擬装置とモニタから構成している。
【0005】
【発明が解決しようとする課題】実時間型機械制御プログラムの開発は、実機（ハードウエア）が試作された後に、プログラムをインストールして、動作確認しながら与えられた仕様を満たすように、エラー&テストを繰り返してる。而して、物理現象や機械動作のシミュレーションモデルが存在するが、開発対象プログラムと直接リンクしてデバイスやテストが行えない。本発明は、シミュレーションモデルと開発プログラムがリンクされた開発環境を作るもので、シミュレーションモデルは、モデルの構築が容易かつ理解しやすい表現にし、開発対象プログラムとの接続が簡単に行なえるようにしたものである。
【0006】
【課題を解決するための手段】請求項1の発明は、プログラムからの制御に基づき制御対象の物理的対象の挙動をシミュレーションするソフトウエア開発支援のシミュレーション装置において、シミュレーションモデルは、個々の物理的対象を表現するデバイスと呼ぶ要素から構成され、デバイスには、デバイスが持つ特性や性質の情報（属性と呼ぶ）、デバイスの取りうる状態とその状態遷移に関する情報、さらに、各状態毎に、その状態が存在を判定する論理的条件（成立条件と呼ぶ）、その状態内で成立する属性の関数的記述（状態関数と呼ぶ）が定義されていることを特徴としたものである。
【0007】請求項2の発明は、デバイス間の属性の依存関係をパイプと呼ぶ機能で実現し、パイプは、あるデバイスの属性と別のデバイスの属性との対応関係とそれらがどのような依存関係にあるかを示す関数的記述から成ることを特徴としたものである。
【0008】請求項3の発明は、ブリッジ部に、シミュレーションモデルの状態変化とプログラムのイベントとの対応関係を定義するテーブルが存在し、該ブリッジ部は、プログラム実行部内で実行中のプログラムが送るデバイスへの指令に基づき、シミュレーション部内のデバイスの状態を遷移させる機能とシミュレーション部内のデバイスの状態変化を検出し、プログラム側にイベント

のディスパッチを行う機能を実行することを特徴としたものである。

【0009】請求項4の発明は、開発支援環境は、シミュレーションモデルがシステムの動作仕様と矛盾する状態を検知してそれを利用者に通知する矛盾検知部を持ち、矛盾検知部は、シミュレーションの実行中に各デバイスの状態を調べて、与えられたシステム動作仕様の記述に基づき、デバイスの異常状態を検出し、それをモニタ上に表示することを特徴としたものである。

【0010】

【発明の実施の形態】

〔開発支援環境全体の構成〕本発明の開発支援環境は、物理的挙動を定義したシミュレータを持ち、開発対象プログラム1とシミュレーションモデル2を同時に動作をさせてシステムの挙動を確認することにより、プログラム上のシステムの動作を決定する各種のパラメータのより正確な値を得るものである。

【0011】〔シミュレーションモデルの構成〕図1は、シミュレーションによる開発支援環境の概略を示す図で、シミュレーションモデル2は、プログラム1からの指令によって制御される機械の挙動を計算機上で模擬するもので、モデルは、デバイスとよぶ構成要素から構成されている。個々のデバイスは、例えば、モータ、アクチュエータ、センサ等の現実の物体と対応しており、物体の挙動はデバイスの内部変数の変化によって表現される。

【0012】デバイスには、デバイスの性質や特性に関する情報を属性として持っており、時間と共に変化する場合には変数になる。モータの属性には、回転速度や回転トルク等の属性がある。属性間の写像関係は、状態関数と呼ぶ関数によって対応づける。属性の値が属性間住時間に依存する場合には時間を含んだ関数になる。

【0013】ところで、実世界の“もの”デバイスは、一般に、ライフサイクルと呼ばれる特定の振る舞いのパターンがあるデバイスの動的な振る舞いパターンを状態遷移として抽象化してモデル化する。各状態毎にその状態内で有効な属性と状態関数を定義する。例えば、静止状態と動作状態では、摩擦に関する法則が異なっている。デバイスの状態遷移を定義するということは、デバイスの挙動を目的をもってモデル化するということになる。一方、状態の中での物理的法則や原理は客観性のある数学的記述を与えている。

【0014】デバイス間の伝達情報（力、温度、電磁気等）の流れは、図2に示すように、パイプと呼ぶ伝達経路を通して、2つのデバイス間で関連し合う属性の値の対応付けを行う。こうしてシミュレーションモデルは、デバイスがパイプによってつながれたネットワーク上の構造になる。

【0015】

〔デバイスの表現〕

```
<Device デバイス名>
  <Attribute 属性>
  <<State 状態名>
    <State Function 状態関数>>*
    <Hold Condition 成立条件>*
    >*
  >*
```

（注　*は<…>を0回以上の繰り返し可）

デバイス名：デバイスの名称

属性　　：デバイスに関連する性質や特性を表現する変数あるいは定数

状態名　　：状態を表現する名称

状態関数　：状態内で成立する法則や原理を表現する属性間の関数的記述表現

成立条件　：状態が成立し続けるための条件、真または偽の2値関数(論理式)で表現

【0016】

〔パイプの表現〕

```
<Pipe
  <Input<デバイス 属性>>
  <Output<デバイス 属性>>
  <Translate 伝立関数>
>*
```

伝達関数　：入力と出力の対応関係の関数的記述

【0017】〔シミュレーションエンジンの構成〕シミュレーションエンジン3は、シミュレーションモデル2を時間の経過に従ってモデル内の状態を更新していき、モデルに挙動を与える、つまり、エンジンによってモデルが活性化されて動作する。すなわち、一定時間毎に、モデル内の全てのデバイスを走査して、デバイスの状態関数に基づき属性の値を更新すると共に成立条件をテストして必要であればデバイスの状態の遷移を起こす。ところで、エンジンに時間を与えるクロックは、プログラム実行部のクロックと同じものを利用する。ただし、クロックは実時間を刻むものでなく、デバグやモニターが容易に行えるように、時間を進める早さを自由に設定が出来る論理的なクロックである。

【0018】ここで、シミュレーションモデルの更新が矛盾なくかつ確実に終了するように、モデル構築において以下の2つの前提を仮定する。

1．デバイス間の伝達にはフィードバックやループは存在しない。

2．デバイス間の伝達には互いに相手からの情報を待ち合うデッドロックは存在しない。

【0019】〔モデル更新の手続き（一定時間間隔（例1msec）で起動される）〕全てのデバイスに対して以下のステップにより時間tにおける属性や状態を計算する。

ステップ1　デバイスの現在状態において、その状態関

数に基づき属性の値を更新する。他のデバイスの影響を受ける属性があり、まだパイプを通して伝達されていない場合には、伝達されるまで待つ。
ステップ2　更新された属性の値から成立条件を判定する。もし条件が満たされなくなっていれば、条件を満たす他の状態に遷移する。他の状態に移れば、そこの状態関数に基づき属性の値を決定する。
ステップ3　更新された属性はパイプを通して他のデバイスに伝えられる。
【0020】〔矛盾検知部の構成〕矛盾検知4は、図3に示すように、シミュレーションモデルの状態が、システムに与えられた仕様と矛盾するあるいは一般的な物理法則などから違反したことを検出（条件判定）する機能である。この機能は、シミュレーションエンジンがシミュレーションモデルの更新を終了すると直ちに起動されて、全てのデバイスの状態がシステムの仕様を満たしているかどうかを判定し、もし満たしていないとそのデバイスをユーザに知らせる。システムの仕様は、システムの挙動について満足すべき条件を表現する論理式の集まりとして、矛盾検知部に格納されている。
【0021】〔矛盾検知の手続き〕シミュレーションモデルの内容が更新される度に以下のステップの手続きを実行する。
ステップ1　システム動作記述より、論理式を取り出す。
ステップ2　取り出した論理式をシミュレーションモデル内のデバイスの属性値や状態の情報に基づき評価して、論理式の成立／不成立を判定する。
ステップ3　判定の結果をモニタやデバガーを通して利用者に通知する。
【0022】〔ブリッジ部の構成〕プログラムの実行はプログラム内の関数が逐次的に処理されるという連続的な手続きの実行である。これに対して、シミュレーションモデルの実行は一定時間毎にモデル内のデバイスの内容を更新していき状態を遷移させるという非連続で状態遷移型の実行である。こうしたそれぞれの実行形態の違いを埋め、互いに情報の通信が行えるようにするのがブリッジ部5の役割である。このブリッジ部の構成を図4に示す。
【0023】ブリッジ部5の動作は、プログラム（図5にオブジェクト指向で記述されたプログラムの概要を示す）から関数の起動（イベントの送信）によりモデルに対して指令を発すると、あるデバイスの状態を別の状態に遷移させる。この関数と状態遷移の対応関係に関する情報は、図7に示すように、状態遷移管理テーブルに登録されている。一方、モデル内であるデバイスの状態が別の状態に遷移することが必要であればプログラムのある関数を起動させる。この状態遷移と起動する関数の対応関係は、図8に示すように、イベント発生管理テーブルに登録されている。シミュレーション実行前にブリッジ部は、この開発環境の利用者が作成したインターフェース定義記述（図6参照）を読み込み、2つのテーブル（状態遷移管理テーブルとイベント発生管理テーブル）を自動的に生成する。図9にプログラムとモデルの通信の流れを示す。
【0024】〔動作例〕以下に、図10乃至図12を参照しながら、複写機における給紙部のコントローラのプログラム開発を例に取り説明する。
〔開発システム〕給紙部は、図10に示すように、モータ10とそれと直結した給紙ローラ11および紙位置検知センサ12と紙搬送ガイド板からなり、これらを制御する制御プログラム13は、モータのオン／オフとセンサの検知情報を受け取る。
【0025】〔開発システムの動作仕様〕図11は、給紙部の動作を説明するための概略図、図12は、給紙プログラムのタイムチャートで、給紙ローラ11は、紙14を送り出し、紙14がセンサ位置12に達する（T1）と、モータ10を停止（T2）し、1msec後に給紙ローラ11を回転させ紙14をレジストローラ15まで送り、さらにレジストローラ15に噛み合わせるためにたわみSを持たせて紙14を送り、モータ10を停止する（T3）。たわみSによって紙14が盛り上がる高さは8mm一定になるようにする。プログラムは、実時間によるシーケンシャルな制御を行うため、決定すべきパラメータは、T1，T2，T3の時刻である。
【0026】〔開発システムのシミュレーションモデル〕シミュレーションモデルに含まれるデバイスは、モータ、給紙ローラ、紙、給紙ガイド板、紙位置センサがあり、それぞれのデバイスモデルの例を、図13乃至図17に順次示す。モータ10は電源がオンになると、一定時間の加速して定常回転に至る。給紙ローラ11は、モータから発生する熱により形状が膨張して半径が大きくなると、回転速度が大きくなる。紙14のたわみは、紙14を縮めた距離とたわむことが可能なガイド板16の隙間幅から、紙14が盛り上がる高さが決まる。こうしたデバイスの属性と取りうる状態を定義し、各状態で成立する物理的法則に基づく属性の関数を決定している。デバイス間の伝達は、モータからローラに回転が伝えられ、またモータから発生した熱量がローラの温度に影響を与える。ローラの回転速度は紙の移動速度へと伝達される。こうした定義はパイプによって定義される。以下に、各デバイスのモデル表現を図的に記述する。
【0027】〔シミュレーションの動作〕シミュレーションモデルでは、そこに与える初期値や設定値（デバイスの属性）によって動作が異なる。従って、例で示したシステムの動作仕様とシミュレーションモデルから、プログラムとシミュレーションモデルを実行するとどのような事象が発生するか、幾つかの場合を挙げてみる。
1．モータの回転時間が長すぎて、レジストローラ前でジャムが発生する。t＝T3にプログラムはモータの回

転を停止させる。しかし、この時シミュレーションモデルの紙デバイスのたわみ高属性の値は、規定の８ｍｍを大きく越えている。矛盾検知部がこの状態が異常であることを検出する。

２．モータの回転時間が短すぎて、紙がレジストローラまで達していない。

【００２８】モータの加速時間を考慮していなかったため、モータをオフにするＴ３の値が早すぎる結果となった。このことは、シミュレーションモデルの紙デバイスの先端位置（ｐｘ１）の値がガイド板の長さＬに満たしていないことにより、矛盾検知部が異常であることを検出する。このように、シミュレーションモデルによりモータの加速／減速時間の考慮やモーラ半径の変化といったより詳細な物理的挙動をシミュレートすることで、プログラムの動作が正しいかどうかを判断することができる。

【００２９】

【発明の効果】

請求項１の効果：物理的対象を状態遷移があるデバイスで表現することにより、システムを目的範的にモデル化することができ、効率がよくまた理解しやすいシミュレーションモデルが得られる。さらに、状態の中に属性の時間的挙動を記述する状態関数によって、数値解析的なシミュレーションの実行が可能であり、実時間制御型プログラムと結合させて、そのデバグや動作確認の開発支援に利用することができる。

【００３０】請求項２の効果：パイプにより、シミュレーションモデルを複数のデバイスのネットワークとして構成することことができ、大規模なシミュレーションモデルを構築することが可能になる。

【００３１】請求項３の効果：プログラム実行部はプログラムを連続的に実行するのに対して、シミュレーション部ではデバイスの状態や属性の値を一定時間間隔で更新していく非連続的な実行形式であり、ブリッジによってこの実行方式の違いを越えて、２つを同時に実行させながら互いの情報の受渡が出来る。

【００３２】請求項４の効果：矛盾検知部により、プログラムの不具合を容易に検出することが出来、開発の効率が向上する。

【図面の簡単な説明】

【図１】　本発明が適用されるシミュレーションによる開発支援環境の概略を示す図である。

【図２】　デバイス間の伝達情報の一例を示す図である。

【図３】　矛盾検知部の構成例を示す図である。

【図４】　ブリッジ部の構成例を示す図である。

【図５】　オブジェクト指向で記述されたプログラムの概要を示す図である。

【図６】　インターフェイスの定義記述を示す図である。

【図７】　状態遷移のための管理テーブルの一例を示す図である。

【図８】　イベント発生のための管理テーブルの一例を示す図である。

【図９】　プログラムとモデル間の通信の流れを示す図である。

【図１０】　複写機の給紙部に関するプログラムと物理的デバイスを示す図である。

【図１１】　給紙部の動作説明をするための概略図である。

【図１２】　給紙部におけるプログラムのタイムチャートの一例を示す図である。

【図１３】　モータのデバイスモデルを示す図である。

【図１４】　給紙ローラのデバイスモデルを示す図である。

【図１５】　紙のデバイスモデルを示す図である。

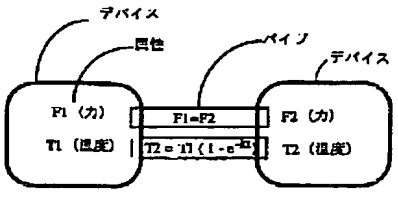
【図１６】　給紙ガイド板のデバイスモデルを示す図である。

【図１７】　センサーのデバイスモデルを示す図である。

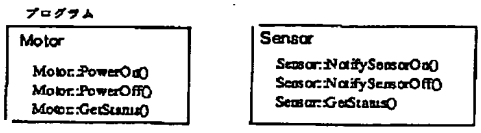
【符号の説明】

1…開発プログラム、2…シミュレーションモデル、3…エンジン部、4…矛盾検出部、5…ブリッジ部。

【図2】



【図5】



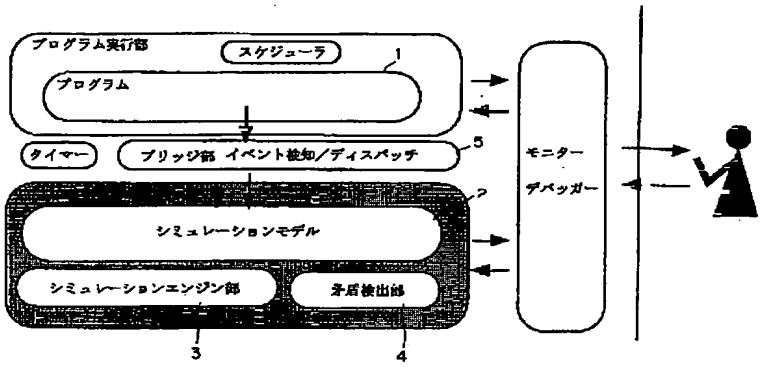
オブジェクト指向で記述されたプログラム概要

【図7】

関数アドレス	状態ID（前）	状態ID（後）
0x00335025	Motor/ST-1	Motor/ST-2
0x00335988	Motor/ST-2	Motor/ST-1
・・・	・・・	・・・

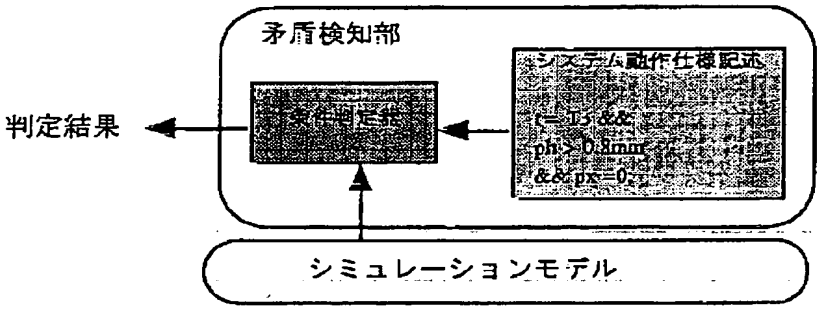
状態遷移のための管理テーブル

【図1】



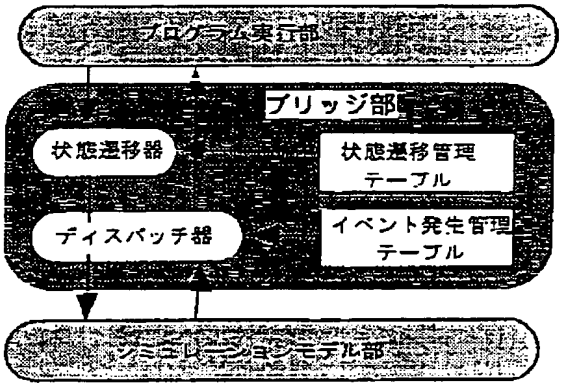
シミュレーションによる開発支援環境の概略

【図3】



矛盾検知部の構成

【図4】



ブリッジ部の構成

【図6】

```
Motor:PowerOn()                    Sensor/NoDetect ->
{                                  Sensor/NoDetect
  Motor/MotorOff ->                {
  Motor/MotorOn                      Sensor::NotifySensorOn();
}                                  }
```

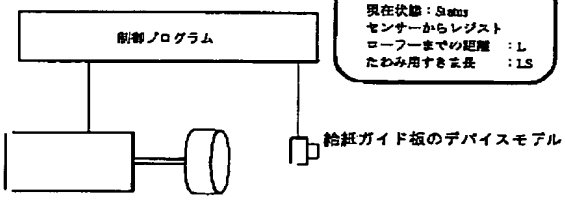
インターフェイス定義記述

【図8】

状態ID（前）	状態ID（後）	関数アドレス
Sensor/ST-2	Sensor/ST-3	0x00335025
Sensor/ST-3	Sensor/ST-2	0x00335988
・・・	・・・	・・・

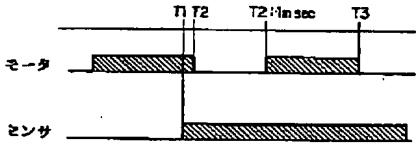
イベント発生のための管理テーブル

【図10】



プログラムと物理的デバイス

【図12】

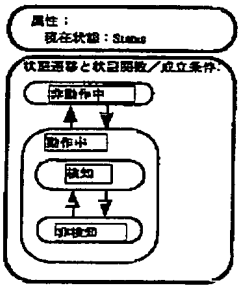


T1　センサ検知
T2　モータ回転停止
T2+1　モータ回転開始
T3　モータ回転停止

プログラムのタイムチャート

【図16】

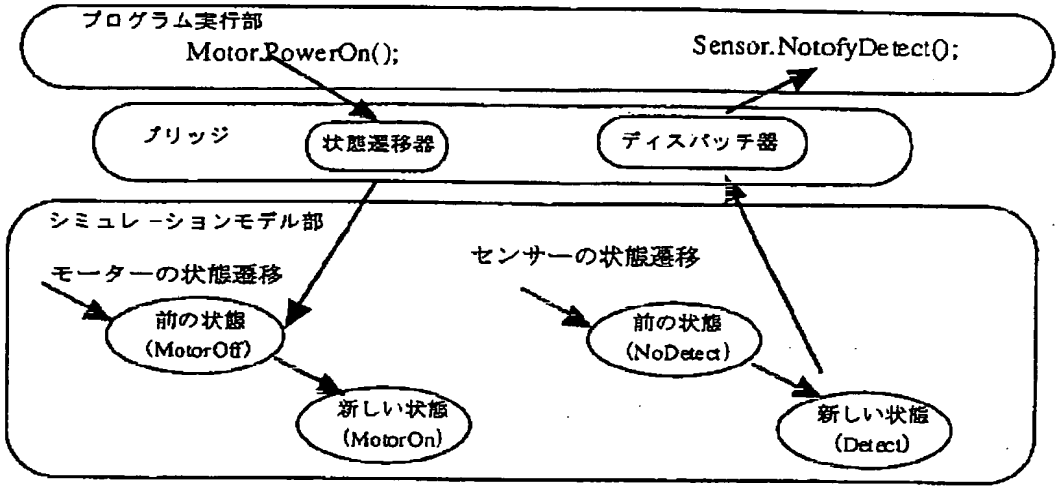
【図17】



センサーのデバイスモデル

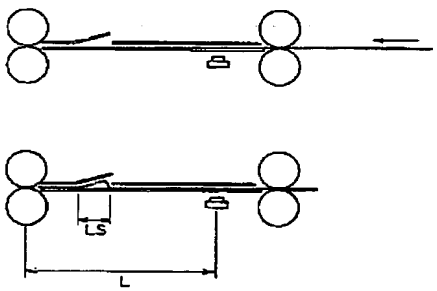
BEST AVAILABLE COPY

【図9】



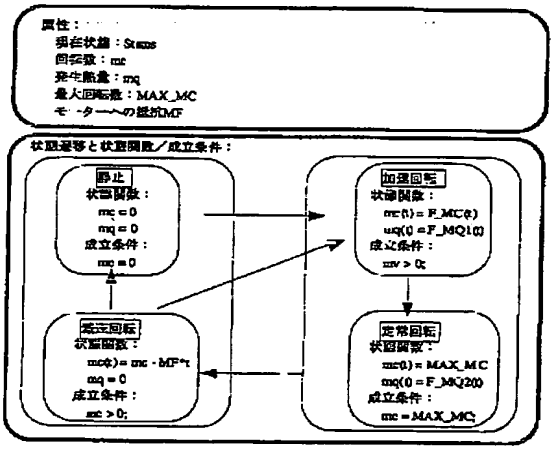
プログラムとモデルの通信の流れ

【図11】



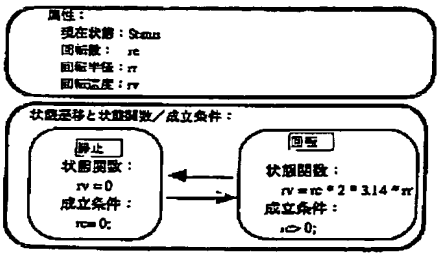
給紙部の動作

【図13】



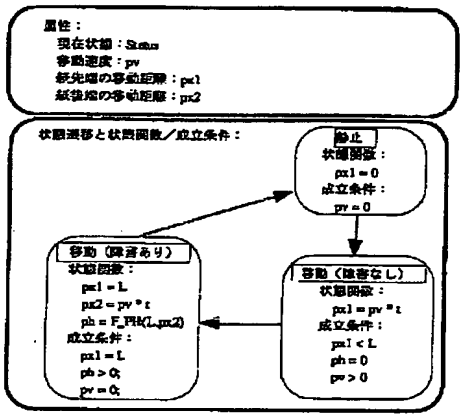
モーターのデバイスモデル

【図14】



給紙ローラーのデバイスモデル

【図15】



旋のデバイスモデル